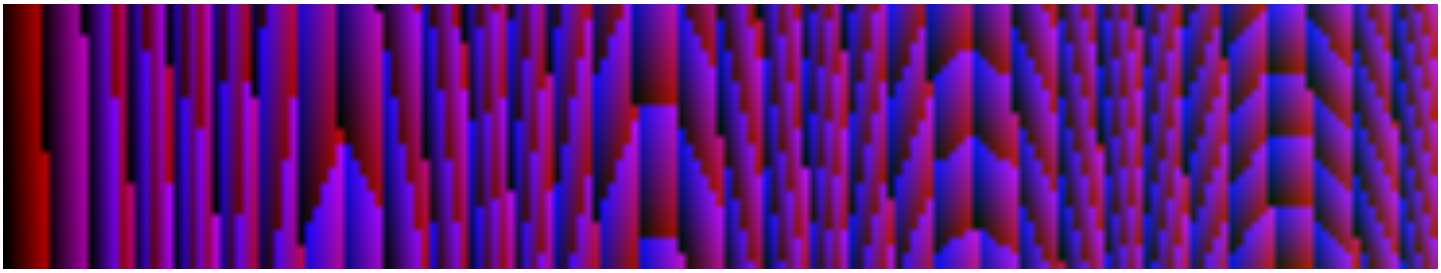
TAKORADI TECHNICAL UNIVERSITY



Portable Bitmap Format

Data Architecture

Prepared for: Research Project on Reduced Data Set Computer.
Project Site: Github.com
Prepared by: Frank Appiah
10/08/2021
Proposal number: 123-4567

EXECUTIVE SUMMARY

Objective

(1) Running ANS Processor: Reduced Data Set Computer (RDSC).
(2) RDSC) Default Array and String Reads.
(3) Setting Appiah Integer Sequence.
(4) Setting Appiah Long Sequence.
(5) Setting Appiah Double Sequence.
(6) Accessing data structures in store[MEM].
(7) (RDSC)::Running successfully.

Goals

(1) (RDSC) Data is information that can be used in making calculations or decisions or sequencing events.
(2) ANSVector Read, ANSQueue Read, ANS-Set Read, ANSList Read, ANS-Formula Read, Arithmetics Read, MEMDraw Read, 5-Divisions Read, Magic No. Read, Floating-point Read, Cardinal Numbers Read, Examine Data Read, Limit Data Read and Data Visualisation Read.

Solution

(1) Offline Data Organisations and Storage(ODOS).
(2) Data Operations and Representation Structures.
(3) Bitmapping data information in portable bitmap format.

Project Outline

- Appiah Number Sequence Storing Program
- This dataware contains numerical values from ANS program.
- Using several data structures like array, list, vector, queue, stack and more.
- Programming in C++.

Main Programming Code

```
/*
 * File:   ApComputeData.cc
 * Author: appiah
 * Appiah Number Sequence Storing Program.
 * Several data structures are used array, list, vector, queue, stack and more.
 * Created on 30 July, 2020 08:47
 */

#include <iostream>
#include <cstdlib>
#include <string>
#include <map>
#include <map>
#include <list>
#include <stack>
#include <queue>
#include <vector>
#include <ctime>
#include <set>
#include <cmath>
#include <stdio.h>
#include <iomanip>
#include <iterator>
#include <sys/unistd.h>
#include <time.h>
#include <sys/ioctl.h>
#include <sched.h>
#include <pthread.h>
#include <fstream>

#define M 3;

namespace{


    typedef struct {
        int row[3][3];
```

```
    int col[3][3];
    int dia[3][3];
}magicer;

std::vector<int> fieldVec;
std::vector<double> sqrtVec, sqrt1Vec, sqrt2Vec, sqrt6Vec;
std::list<int> ansList;
std::queue<int> ansQueue;
std::vector<int> ansVector;
std::stack<int> anStack;
std::set<int> ansSet;
std::map<int, int> ansMap;
const int m=13, n=6;
int datagrid[m][n];
int go=1;
//tree
int const enumSize =78;
std::string ansString;
int ansi[]={-2025, -945, -665, -504, -405, -342, -297, -225, -214, 0, 1, 3, 4, 5, 8, 9, 10, 12, 15, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40,  45, 51, 52, 54, 56, 57, 59, 60, 63, 65, 66, 67, 72, 75, 78, 81, 85, 89, 90, 100, 105, 120, 125, 135, 145, 165, 180, 185, 218, 225, 270, 300, 315, 345, 360, 405, 505, 665, 950, 2025};
std::string ansiCardinal[]={"minus two thousand and twenty five", "minus nine hundred and forty five", "minus six hundred and sixty five", "minus five hundred and four", "minus four hundred and five", "minus three hundred and forty two", "minus two hundred and ninety seven", "minus two hundred and twenty five", "minus two hundred fourteen", "zero", "one", "three", "four", "five", "eight", "nine", "ten", "twelve", "fifteen", "sixteen", "seventeen", "eighteen", "twenty", "twenty one", "twenty three", "twenty four", "twenty five", "twenty seven", "twenty eight", "thirty", "thirty one", "thirty two" , "thirty three", "thirty four", "thirty five", "thirty six", "thirty eight", "thirty nine", "forty",  "forty five", "fifty", "fifty two", "fifty four", "fifty six", "fifty seven", "fifty nine", "sixty", "sixty three", "sixty five", "sixty six", "sixty seven", "seventy", "seventy five", "seventy eight", "eighty one", "eighty five", "eighty nine", "ninety", "hundred", "hundred and five", "hundred and twenty", "hundred and twenty five", "hundred and thirty five", "hundred and forty five", "hundred and sixty five", "hundred and eighty", "hundred and eighty five", "two hundred and eighty five", "two hundred and twenty five", "two hundred and seventy", "three hundred", "three hundred and fifteen", "three hundred and forty five", "three hundred and sixty", "four hundred and five", "five hundred and five", "six hundred and sixty five", "nine hundred and fifty", "two thousand and twenty five"};
int divs[]={1, 5, 9, 13, 17};
int res[]={0, 0};
long* ansl;
magicer magic;
```

```
double* ansd;

int getGO(){
    return go;
}

std::string enterStarts(){
    std::string start;
    std::cout<< std::endl<< "Enter[ok] to start>>"<< std::endl;
    getline(std::cin, start);
    return start;
}

std::string enterContinues(){
    std::string con="go";
    std::cout<< std::endl<< "Enter number[go(1)/no(2)] to continue>>"<< std::endl;
    getline(std::cin, con);
    go=strtol(con.c_str(), 0, 10);
    std::cout << "Go Value:=#" << go << std::endl;
    return con;
}


int* getFunctValues(std::string instr){
    std::cout<<instr;
    std::string value="";
    std::string apostr;
    std::string bpostr;

    getline(std::cin, value);
    //char* v=value.c_str();

    int apos=value.find(",");
    apostr=value.substr(0, apos);
    int v1=strtol(apostr.c_str(), 0, 10);
    res[0]=v1;
    std::cout<<"a:=:"<<res[0];

    bpostr=value.substr(apos+1, value.length()-1);
```

```
    int v2=strtol(bpostr.c_str(), 0, 10);
    res[1]=v2;
    std::cout<<"  b:=:"<<res[1];
    return res;
  }

  void examineData(){
    std::cout <<std::endl<< "----Examine Data----" << std::endl;
    enterStarts();
    std::cout <<std::endl<< "-------------------------------------------------" << std::endl;
    std::cout <<std::endl<< "Count        Data        Probable    Prob. Func" << std::endl;
    std::cout <<std::endl<< "-------------------------------------------------" << std::endl;
    enterStarts();
    int count=0;
    for(int i=0;i<enumSize;i+=1){
      //if(ansi[i]>0 && ansi[i+1]>0){
      std::string probe;
      div_t re=std::div(1, ansi[i+1]-ansi[i]);
      float valr=atof("0.0"+re.rem/2);
      valr=valr+re.quot;
      probe=(valr)>0.5? "1":"0";
      if(valr>0.5) count++;
      std::cout<<i<<"          "<<ansi[i+1]<<" , "<<ansi[i]<<"         "<<probe<<"
"<<std::fixed<<re.quot<<".0"<<re.rem/2<<std::endl;
      //       } else std::cout<<"neg encounter"<<std::endl;
      if(i%5==0)  {enterContinues(); std::cout << std::endl; }
      if(getGO()>1) break; else continue;
    }
    sleep(1);
    std::cout<< "--Probability Density Function(pdf)"<<std::endl;
    enterStarts();
    int* vals;
    vals=getFunctValues("Enter the range values Format:[a,b]>>");
    // float res=0;
    int r1=(vals[1]-vals[0]);std::cout<<" diff::="<<r1;
    div_t res=std::div(r1, 2025*2);

    std::cout<<"   pdf:="<<(res.quot)<<".0"<<std::fixed<<res.rem/2<<std::endl<<std::endl;
    sleep(1);
    std::cout<< "--Probability Distribution Function(PDF)"<<std::endl;
```

```
    enterStarts();
    vals=getFunctValues("Enter the range values Format:[a,b]>>");
    r1=(vals[1]-vals[0]);std::cout<<" diff::="<<r1;
    int r2=(2025-vals[0]);
    res=std::div(r1, r2);
    std::cout<<"   PDF:="<<res.quot<<".0"<<res.rem/2<<std::endl;

    enterContinues();
}

void getCardinalNumbers(){
    std::cout <<std::endl<< "----Cardinal Numbers----" << std::endl;
    enterStarts();
    std::cout <<std::endl<< "-------------------------------------------------" << std::endl;
    std::cout <<std::endl<< "Number         Cardinal" << std::endl;
    std::cout <<std::endl<< "-------------------------------------------------" << std::endl;

    for(int i=0;i<enumSize;i++){
        std::cout<< ansi[i]<<"          Amount:=:"<<ansiCardinal[i]<<std::endl;
        if(i%5==0)  {enterContinues(); std::cout << std::endl; }
        if(getGO()>1) break; else continue;
    }
    sleep(1);
    std::cout << "1099886703552000          " << "Amount:=:One thousand and ninety nine trillion eight hundred and eighty six billion seven hundred and three million five hundred and fifty two thousand"<<std::endl;
    std::cout << "131382799891200           " << "Amount:=:One hundred and thirty one trillion three hundred and eighty two billion seven hundred and ninety nine million eight hundred and ninety one thousand and two hundred"<<std::endl;
    sleep(1);
    std::cout << "3170893824000             " << "Amount:=:Three trillion one hundred and seventy billion eight hundred and ninety three million eight hundred and twenty four thousand"<<std::endl;
    std::cout << "38661097149675            " << "Amount:=:Thirty eight trillion six hundred and sixty one billion ninety seven million one hundred forty nine thousand six hundred and seventy five"<<std::endl;

    enterContinues();
}

void setMagicNumber(){
    magic.row[0][0]=18;
    magic.row[0][1]=12;
```

```
    magic.row[0][2]=15;

    magic.row[1][0]=11;
    magic.row[1][1]=14;
    magic.row[1][2]=20;

    magic.row[2][0]=16;
    magic.row[2][1]=19;
    magic.row[2][2]=10;

    magic.col[0][0]=magic.row[0][0];
    magic.col[0][1]=magic.row[1][0];
    magic.col[0][2]=magic.row[2][0];

    magic.col[1][0]=magic.row[0][1];
    magic.col[1][1]=magic.row[1][1];
    magic.col[1][2]=magic.row[2][1];

    magic.col[2][0]=magic.row[0][2];
    magic.col[2][1]=magic.row[1][2];
    magic.col[2][2]=magic.row[2][2];

    magic.dia[0][2]=magic.row[0][2];
    magic.dia[1][1]=magic.row[1][1];
    magic.dia[2][0]=magic.row[2][0];

}

/*
 *Display magic number 45 row X col
 */
void readMN45(){
    std::cout<<std::endl<<"--Magic Number 45--" <<std::endl;
    enterStarts();
    std::cout << "---------------"<<std::endl;
    for(int i=0;i<3;i++){
        {
            std::cout<<"+";
            for(int j=0;j<3;j++){
                std::cout << "|"<< magic.row[i][j]<<"|";
```

```
        }
        std::cout<< "=45|"<<std::endl << "----------"<<std::endl;
        sleep(1);
    }
}
std::cout<<"=|45|" << "|45|"<< "|45|45/45"<<std::endl;
std::cout<<std::endl<< "----Z-Magic Numbers" <<std::endl;
enterStarts();
for(int i=0;i<3;i++){
    {
        std::cout<<"+";
        for(int j=0;j<3;j++){
            if((i==1 && j==0) ||(i==1 &&j==2))
                std::cout << "|0 |";
            else
                std::cout << "|"<< magic.row[i][j]<<"|";
        }
        if(i==1)
            std::cout<< "14|"<<std::endl << "----------"<<std::endl;
        else
            std::cout<< "45|"<<std::endl << "----------"<<std::endl;
        sleep(1);  }

} std::cout<<"=|34|" << "|45|"<< "|25|104/135=0.77037037-----(New Number::(104)"<<std::endl;
std::cout<<std::endl<< "----R-Magic Numbers" <<std::endl;
enterStarts();
for(int i=0;i<3;i++){
    {
        std::cout<<"+";
        for(int j=0;j<3;j++){
            if((i==0 && j==2) ||(i==1 &&j==2) ||(i==2 &&j==1))
                std::cout << "|0 |";
            else
                std::cout << "|"<< magic.row[i][j]<<"|";
            sleep(1);
        }
        if(i==0)
            std::cout<< "30|"<<std::endl << "----------"<<std::endl;

        if(i==1)
```

```
            std::cout<< "25|"<<std::endl << "---------------"<<std::endl;
        if(i==2)
            std::cout<< "45|"<<std::endl << "---------------"<<std::endl;
    }

} std::cout<<"=|45|" << "|26|"<< "|10|81/100=0.81-----(New Number::(26)"<<std::endl;
std::cout<<std::endl<< "----W-Magic Numbers" <<std::endl;
enterStarts();
for(int i=0;i<3;i++){
    {
        std::cout<<"+";
        for(int j=0;j<3;j++){
            if((i==0 && j==1) ||(i==2 &&j==1))
                std::cout << "|0 |";
            else
                std::cout << "|"<< magic.row[i][j]<<"|";
            sleep(1);
        }
        if(i==1)
            std::cout<< "14|"<<std::endl << "---------------"<<std::endl;
        else
            std::cout<< "45|"<<std::endl << "---------------"<<std::endl;
    }

} std::cout<<"=|45|" << "|14|"<< "|25|84/84=1-----(New Number::(84)"<<std::endl;
std::cout<<std::endl<< "----N-Magic Numbers" <<std::endl;
enterStarts();
for(int i=0;i<3;i++){
    {
        std::cout<<"+";
        for(int j=0;j<3;j++){
            if((i==0 && j==1) ||(i==2 &&j==1))
                std::cout << "|0 |";
            else
                std::cout << "|"<< magic.row[i][j]<<"|";
            sleep(1);
        }
        if(i==1)
            std::cout<< "14|"<<std::endl << "---------------"<<std::endl;
        else
```

```
                std::cout<< "45|"<<std::endl << "---------------"<<std::endl;
            }

        } std::cout<<"=|45|" << "|14|"<< "|45|84/84=1-----(New Numbers:: (84)"<<std::endl;

        enterContinues();
    }

    void uninitialise(){
        std::cout << "Un-initializing data store for all structures....." << std::endl << std::endl;
        enterStarts();
        int i=0;
        for(i=0; i<enumSize; i++) {
            ansi[i]=0;
        }
        enterContinues();
    }


    void setIntANS(){
        std::cout << "Initializing Appiah Number Sequence :Size(" << enumSize+4 << ")" << std::endl;
    }

    int* getIntANS(){
        return ansi;
    }

    void setANString(){
        std::cout << "Building the ANS String Data[Starts]"<< std::endl;
        enterStarts();
        std::string value;

//        for(int k=0;k<enumSize;k++){
//            value="";
//            const char* v='';
//            std::__convert_to_v(v, getIntANS()[k], 0, NULL);
//            std::cout << "convert::"<< v << std::endl;
//            value=v;
//            if(k%6==0)
//            { value+="/n";}
```

```
//            //// value=v;
//             ansString.append(value);
//             std::cout << "Data [" << k << "] is appended into string store." << std::endl;
//        }
//        std::cout << "Building the ANS String Data[Ends]"<< std::endl;
        enterContinues();
    }


    void setANSVector(){
        std::cout << "Building the ANS Vector Data[Starts]"<< std::endl;
        enterStarts();
        for(int k=0;k<enumSize;k++){
            ansVector.push_back(getIntANS()[k]);
            std::cout << "Data ["<< k << "] is pushed into vector store." << std::endl;
        }
        std::cout << "Building the ANS Vector Data[Ends]"<< std::endl;
        enterContinues();
    }

    void readANSVector(){

        std::cout << std::endl<< "Reading the ANS Vector Data[Starts::MEMOVEC]:"<< ansVector.size()<<
std::endl;
        enterStarts();
        std::string res;
        for(int k=0; k<enumSize;k++){
            std::cout << "Datactor ["<< k << "]:=:" << ansVector.at(k) << " ";

            if(k%3==0)  {enterContinues(); std::cout << std::endl; }
            if(getGO()>1) break; else continue;
        }
        std::cout << "Datactor ["<< 78 << "]:=:" << "1099886703552000" << std::endl;
        sleep(1);
        std::cout << "Datactor ["<< 79 << "]:=:" << "131382799891200" << " ";
        std::cout << "Datactor ["<< 80 << "]:=:" << "3170893824000" << std::endl;
        sleep(1);
        std::cout << "Datactor ["<< 81 << "]:=:" << "38661097149675" << " ";
        std::cout <<std::endl<< "Reading the ANS Vector Data[Ends::MEMOVEC]"<< std::endl;
```

```
        enterContinues();
    }

    void setANSQueue(){
        std::cout <<  std::endl<< "Building the ANS Queue Data[Starts]"<< std::endl;
        enterStarts();
        for(int k=0;k<enumSize;k++){
            ansQueue.push(getIntANS()[k]);
            std::cout << "Data ["<< k << "] is pushed into queue store." << std::endl;
        }
        std::cout << "Building the ANS Queue Data[Ends]"<< std::endl;
        enterContinues();
    }

    void readANSQueue(){

        std::cout <<  std::endl<< "Reading the ANS Queue Data[Starts::MEMOQUE]:"<< ansQueue.size() <<
std::endl;
        enterStarts();
        int k=0;
        std::cout<< "Which kind of queue? (1) FIFO (2)LIFO (3)FIRO>>" <<std::endl;
        std::string queue;
        getline(std::cin, queue);
        int sel=strtol(queue.c_str(), 0, 10);
        if(sel==1){
            std::cout<< "--Queue FIFO-Disposition" <<std::endl;

            while(!ansQueue.empty()){
                std::cout << "Data ["<< k++ << "]=:=" << ansQueue.front()  << "    ";
                ansQueue.pop();

                if(k%3==0) { enterContinues(); std::cout << std::endl;}
                if(getGO()>1) break; else continue;
            }
            std::cout << "Data ["<< 78 << "]=:=" << "1099886703552000" << std::endl;
            std::cout << "Data ["<< 79 << "]=:=" << "131382799891200" << " ";
            std::cout << "Data ["<< 80 << "]=:=" << "3170893824000" << std::endl;
            std::cout << "Data ["<< 81 << "]=:=" << "38661097149675" << " ";
        }
        else if(sel==2){
```

```
        std::cout<< "--Queue LIFO-Disposition" <<std::endl;
        k=enumSize;

        std::cout << "Data ["<< 81 << "]=:=" << "38661097149675" << " ";
        std::cout << "Data ["<< 80 << "]=:=" << "3170893824000" << std::endl;
        std::cout << "Data ["<< 79 << "]=:=" << "131382799891200" << " ";
        std::cout << "Data ["<< 78 << "]=:=" << "1099886703552000" << std::endl;

        while(k>0){
            std::cout << "Data ["<< k-- << "]=:=" << ansi[k]  << "    ";
            ansQueue.pop();

            if(k%3==0) { enterContinues(); std::cout <<std::endl;}
            if(getGO()>1) break; else continue;
        }
    }
    else if(sel==3){
        std::cout<< "--Queue FIRO-Disposition" <<std::endl;
        k=enumSize;
        //unsigned int randv[enumSize]={1, 4, 5, 6, 78, 12, 8, 9, 11, 22, 33, 44, 55, 66, 77, 74, 70, 71, 18, 14,
27, 66, 60, 45, 50, 56, 57, 59, 52, 21};
        unsigned int randv[2]={1, 30};

        srandom(randv[1]);
        k=random();
        std::cout << "Data ["<< k << "]=:=" << ansi[k]  << "    ";
        ansQueue.pop();


        std::cout << "Data ["<< 81 << "]=:=" << "38661097149675" << " ";
        std::cout << "Data ["<< 80 << "]=:=" << "3170893824000" << std::endl;

        std::cout << "Data ["<< 79 << "]=:=" << "131382799891200" << " ";
        std::cout << "Data ["<< 78 << "]=:=" << "1099886703552000" << std::endl;

    }

    std::cout << "Reading the ANS Queue Data[Ends::MEMOQUE]"<< std::endl;
    setANSQueue();
    enterContinues();
```

```
    }

    void setANSMap(){
        std::cout <<  std::endl<< "Building the ANS Map Data[Starts]"<< std::endl;
        enterStarts();
        for(int k=0;k<enumSize;k++){

            // ansMap.insert(k, getIntANS()[k]);
            std::cout << "Data ["<< k << "] is inserted into map store." << std::endl;
        }
        std::cout << "Building the ANS Map Data[Ends]"<< std::endl;
        enterContinues();
    }

    void setANSet(){
        std::cout <<  std::endl<< "Building the ANS Set Data[Starts]"<< std::endl;
        enterStarts();
        for(int k=0;k<enumSize;k++){
            ansSet.insert(getIntANS()[k]);
            std::cout << "Data ["<< k << "] is inserted into set store." << std::endl;
        }
        std::cout << "Building the ANS Set Data[Ends]"<< std::endl;
        enterContinues();
    }

    void readANSet(){

        std::cout << std::endl<< "Reading the ANS Dataset[Starts::MEMOSET]:"<< ansSet.size() << std::endl;
        enterStarts();
        int k=0;
        //  auto std::iterator<> it=ansSet.begin();
//          while(it !=ansSet.end()){
//           std::cout << "Data ["<< k++ << "]=:="  << *it << std::endl;
//                if(k%3==0) std::cout << endl;
//           ++it;
//          }
//         std::cout << "Reading the ANS Dataset[Ends::MEMOSET]"<< std::endl;
        enterContinues();
    }
```

```
void setANSList(){
    std::cout << std::endl<< "Building the ANS List Data[Starts]"<< std::endl;
    enterStarts();
    for(int k=0;k<enumSize;k++){
        ansList.push_back(getIntANS()[k]);
        std::cout << "Data ["<< k << "] is pushed into list store." << std::endl;
    }
    std::cout << "Building the ANS List Data[Ends]"<< std::endl;
    enterContinues();
}

void readANSList(){

    std::cout << std::endl<< "Reading the ANS Datalist[Starts::MEMOLIST]:"<< ansList.size() << std::endl;
    enterStarts();
    int k=0;

    while(!ansList.empty()){
        std::cout << "Data ["<< k++ << "]<=>" << ansList.front()  << "    ";
        ansList.pop_front();
        if(k%3==0) std::cout << std::endl;
    }

    std::cout << "Data ["<< 78 << "]<=>" << "1099886703552000";
    std::cout << "Data ["<< 79 << "]<=>" << "131382799891200" << " ";

    std::cout << "Data ["<< 80 << "]<=>" << "3170893824000" << std::endl;
    std::cout << "Data ["<< 81 << "]<=>" << "38661097149675" << " ";

    std::cout << "Reading the ANS Datalist[Ends::MEMOLIST]"<< std::endl;
    setANSList();
    enterContinues();
}

long* getLongANS(){
    return ansl;
}

void setDoubleANS(){
    // ansd[0]= 1099886703552000;
```

```
    // ansd[1]= 131382799891200;
    //  ansd[2]= 3170893824000;
    //  ansd[3]= 38661097149675;
  }

  double* getDoubleANS(){
    return ansd;
  }

  int getEnumSize(){
    return enumSize;
  }

  std::string getANString() {
    return ansString;
  }

  void read5Divs(){
    std::cout <<std::endl<< "  (RDSC) 5-Division Representation  " << std::endl;
    std::cout << "---------------------------------------"<< std::endl;

    enterStarts();

    for(int i=0;i<5;i++) {
      std::cout << i+1 << ":: (RDSC) total_"<<i<<":="<< divs[i]<< std::endl;
    }
    std::cout<< std::endl << "Division Member Data set"<<std::endl<<
    "---------------------------------------"<< std::endl;

    for(int i=0;i<1;i++) {
      std::cout << "(RDSC) Member 1:={"<< divs[i]<<"},{"<<divs[i+1]<<"}"<<",{" <<divs[i+2]<< "},
{"<<divs[i+3]<<"},{"<<divs[i+4]<<"}"<<std::endl;
      std::cout << "(RDSC) Member 2:={"<< divs[i]<<","<<divs[i+1]<<"},{"<<divs[i]<<","<<divs[i+2]<<"}"<<",
{" <<divs[i]<<","<<divs[i+3]<< "},{"<<divs[i]<<","<<divs[i+4]<<"},";
      std::cout << "{"<< divs[i+1]<<","<<divs[i+2]<<"},{"<<divs[i+1]<<","<<divs[i+3]<<"}"<<",{"
<<divs[i+1]<<","<<divs[i+4]<< "}"<<std::endl;
      std::cout << ",{"<< divs[i+2]<<","<<divs[i+3]<<"},{"<<divs[i+2]<<","<<divs[i+4]<<"}"<<",{"
<<divs[i+3]<<","<<divs[i+4]<< "}"<<std::endl;
      std::cout << "(RDSC) Member 3:={"<< divs[i]<<","<<divs[i+1]<<","<<divs[i+2]<<"},
{"<<divs[i]<<","<<divs[i+1]<<","<<divs[i+4]<<"}";
```

```
            std::cout <<",{"<< divs[i]<<","<<divs[i+2]<<","<<divs[i+3]<<"},
{"<<divs[i]<<","<<divs[i+2]<<","<<divs[i+4]<<"}";
            std::cout <<",{"<< divs[i]<<","<<divs[i+1]<<","<<divs[i+3]<<"},
{"<<divs[i+1]<<","<<divs[i+2]<<","<<divs[i+3]<<"},{"<<divs[i+1]<<","<<divs[i+2]<<","<<divs[i+4]<<"}";
            std::cout <<",{"<< divs[i+1]<<","<<divs[i+3]<<","<<divs[i+4]<<"},
{"<<divs[i+2]<<","<<divs[i+3]<<","<<divs[i+4]<<"}"<<std::endl;
            std::cout << "(RDSC) Member 4:={"<< divs[i]<<","<<divs[i+1]<<","<<divs[i+2]<<","<<divs[i+3]<<"},
{"<<divs[i]<<","<<divs[i+2]<<","<<divs[i+3]<<","<<divs[i+4]<<"}"<<std::endl;
            std::cout << "(RDSC) Member 5:={"<<
divs[i]<<","<<divs[i+1]<<","<<divs[i+2]<<","<<divs[i+3]<<","<<divs[i+4]<<"}"<<std::endl;
        }
        enterContinues(); std::cout << std::endl;
    }

    void formInstr(){
        std::cout <<std::endl<< "================================" << std::endl;
        std::cout << "  (RDSC) Formula Set" << std::endl;
        std::cout << "================================" << std::endl;
        enterStarts();
        std::cout << "(RDSC) max n          Formula" << std::endl;
        std::cout << "--------------------------------" << std::endl;
        std::cout << "(RDSC)  1             45n" << std::endl;
        std::cout << "(RDSC)  2             20n+5" << std::endl;
        std::cout << "(RDSC)  3             15n" << std::endl;
        std::cout << "(RDSC)  4             11n+1" << std::endl;
        std::cout << "(RDSC)  5             9n" << std::endl;
        std::cout << "(RDSC)  6             7n+3" << std::endl;
        std::cout << "(RDSC)  7             6n+3" << std::endl;
        std::cout << "(RDSC)  8             5n+5" << std::endl;
        std::cout << "(RDSC)  9             5n" << std::endl;
        std::cout << "(RDSC)  10            1+n^2+n^6+n^7" << std::endl;

    }

    void readFormula() {
        formInstr();
        int sel=1, i=5, cal=1;
        int n=1;

        std::string value;
```

```
std::cout << "Select a formula from set(1-10)>>";
getline(std::cin, value);
sel=strtol(value.c_str(), 0, 10);

std::string val;
std::cout << std::endl<< "Which type of calculation? n-value(1) or generate(2)>>";
getline(std::cin, val);
cal=strtol(val.c_str(), 0, 10);

if(cal==1){
    std::string value1;
    std::cout << std::endl<< "Enter n value in formula>>";
    getline(std::cin, value1);
    n=strtol(value1.c_str(), 0, 10);
    int64_t res=0;
    switch(sel){
        case 1:
            res=45*n;
            break;
        case 2:
            res=20*n+5;
            break;
        case 3:
            res=15*n;
            break;
        case 4:
            res=11*n+1;
            break;
        case 5:
            res=9*n;
            break;
        case 6:
            res=7*n+3;
            break;
        case 7:
            res=6*n+3;
            break;
        case 8:
            res=5*n+5;
            break;
```

```
        case 9:
            res=5*n;
            break;
        case 10:
            res=1+pow((double)n, (double)2.0)+ pow((double)n, (double)6.0)+pow((double)n, (double)7.0);
        default:
            std::cout << "Selection cannot be found"<< std::endl;
            break;
    }
    std::cout << std::endl << "Result of Formula "<<sel <<" ::="<< res;
    enterContinues();

}
else {
    std::cout << std::endl<< "Enter the number of iterations>>";
    std::string vale;
    getline(std::cin, vale);
    i=strtol(vale.c_str(), 0, 10);
    int64_t result[i];
    enterStarts();
    switch(sel){
        case 1:
            for(int l=0;l<=i;l++) {
                result[l]=45*l;
            }
            break;
        case 2:
            for(int l=0;l<=i;l++) {
                result[l]=20*l+5;
            }

            break;
        case 3:
            for(int l=0;l<=i;l++) {
                result[l]=15*l;
            }

            break;
        case 4:
            for(int l=0;l<=i;l++) {
```

```
        result[l]=11*l+1;
    }

    break;
case 5:
    for(int l=0;l<=i;l++) {
        result[l]=9*l;
    }

    break;
case 6:
    for(int l=0;l<=i;l++) {
        result[l]=7*l+3;
    }

    break;
case 7:
    for(int l=0;l<=i;l++) {
        result[l]=6*l+3;
    }

    break;
case 8:
    for(int l=0;l<=i;l++) {
        result[l]=5*n+5;
    }

    break;
case 9:
    for(int l=0;l<=i;l++) {
        result[l]=5*l;
    }
    break;
case 10:
    std::cout<<std::endl<<"--Missing Number Calculations--" << std::endl;
    for(int l=0;l<=i;l++) {
        result[l]=1+pow(l, 2.0)+ pow(l, 6.0)+pow(l, 7.0);
    }
    break;
default:
```

```
            std::cout << "Selection cannot be found"<< std::endl;
            break;

        }
        std::cout <<std::endl<< "     Iterations on Formula "<< sel << std::endl;
        std::cout << "================================="<< std::endl;
        enterStarts();
        std::cout << "n           result"<< std::endl;
        for(int p=0;p<i;p++){
           std::cout <<""<<p<<"             "<<result[p]<< std::endl;
           if(p%10==0){
              enterContinues();
              if(getGO()>1) break; else continue;
           }
        }
     }
  }

  void formArith(){
     std::cout <<std::endl<< "================================" << std::endl;
     std::cout << "  (RDSC) Arithmetics Set" << std::endl;
     std::cout << "================================" << std::endl;
     enterStarts();
     std::cout << "(RDSC) opcode n        operation" << std::endl;
     std::cout << "--------------------------------" << std::endl;

     std::cout << "(RDSC)  1            add:: +" << std::endl;
     std::cout << "(RDSC)  2            sub:: -" << std::endl;

     std::cout << "(RDSC)  3            mul:: *" << std::endl;
     std::cout << "(RDSC)  4            div:: /" << std::endl;

     std::cout << "(RDSC)  5            rem:: %" << std::endl;
     std::cout << "(RDSC)  6            and:: &" << std::endl;
//      std::cout << "(RDSC)  7            not:: !" << std::endl;
//      std::cout << "(RDSC)  8            gt :: >" << std::endl;
//      std::cout << "(RDSC)  9            lt :: <" << std::endl;
//      std::cout << "(RDSC)  10           eq :: =" << std::endl;
//      std::cout << "(RDSC)  11           gte:: >=" << std::endl;
//      std::cout << "(RDSC)  12           leq:: <=" << std::endl;
```

```
//        std::cout << "(RDSC)  13          neq:: <>" << std::endl;
//        std::cout << "(RDSC)  14          pow:: ^" << std::endl;
//        std::cout << "(RDSC)  15          lsh:: >>" << std::endl;
//        std::cout << "(RDSC)  16          rsh:: <<" << std::endl;
//
    }

    void readArith(){
        std::cout<<std::endl<< "--Performing Arithmetic Calculations--"<<std::endl;
        formInstr();
        int sel=1, i=5, cal=1;
        int n=1;

        std::string value;
        std::cout << "Select a formula from set(1-10)>>";
        getline(std::cin, value);
        sel=strtol(value.c_str(), 0, 10);

        std::cout << std::endl<< "Enter the number of iterations>>";
        std::string vale;
        getline(std::cin, vale);
        i=strtol(vale.c_str(), 0, 10);
        int64_t result[i];
        enterStarts();
        switch(sel){
            case 1:
                for(int l=0;l<=i;l++) {
                    result[l]=45*l;
                }
                break;
            case 2:
                for(int l=0;l<=i;l++) {
                    result[l]=20*l+5;
                }

                break;
            case 3:
                for(int l=0;l<=i;l++) {
                    result[l]=15*l;
                }
```

```
    break;
case 4:
    for(int l=0;l<=i;l++) {
        result[l]=11*l+1;
    }

    break;
case 5:
    for(int l=0;l<=i;l++) {
        result[l]=9*l;
    }

    break;
case 6:
    for(int l=0;l<=i;l++) {
        result[l]=7*l+3;
    }
    break;
case 7:
    for(int l=0;l<=i;l++) {
        result[l]=6*l+3;
    }

    break;
case 8:
    for(int l=0;l<=i;l++) {
        result[l]=5*l+5;
    }

    break;
case 9:
    for(int l=0;l<=i;l++) {
        result[l]=5*l;
    }
    break;
case 10:
{ std::cout<<std::endl<<"--Missing Number Calculations--" << std::endl;
  for(int l=0;l<=i;l++) {
      result[l]=1 + pow(l, 2.0) + pow(l, 6.0) + pow(l, 7.0);
```

```
      }
    }   break;
    default:
        std::cout << "Selection cannot be found"<< std::endl;
        break;
}
formArith();

std::string valop;
int opcode;
std::cout << "Select an operation from set(1-6)>>";
getline(std::cin, valop);
opcode=strtol(valop.c_str(), 0, 10);

int32_t opResult[i];
enterStarts();
std::cout <<std::endl<< "   Integer Iterations on Formula "<< sel << std::endl;
std::cout << "================================="<< std::endl;
std::cout << "n            result"<< std::endl;
for(int p=0;p<i;p++){
    std::cout <<""<<p<<"              "<<result[p]<< std::endl;
    if(p%10==0){
        enterContinues();
        if(getGO()>1) break; else continue;
    }
}
std::cout << "================================="<< std::endl;

std::cout<< "Arithmetic Opcode:: "<< opcode <<" has operation =:=";
switch(opcode){
    case 1:
    {int64_t sum=0;
     for(int u=0;u<i;u++){
         sum+=result[u];
     }
     std::cout<< sum <<std::endl <<std::endl;
    }  break;
    case 2:
    { int64_t sub=result[0];
      for(int u=1;u<i;u++){
```

```
            sub-=result[u];
          }
          std::cout<< sub <<std::endl <<std::endl;
        }   break;
        case 3:
        {   float_t div=result[0];
            for(int u=0;u<i;u++){
                div/=result[u];
            }
            std::cout<< std::fixed <<div <<std::endl <<std::endl;
        }   break;
        case 4:
        {   int64_t prod=1;
            for(int u=0;u<i;u++){
                prod*=result[u];
            }
            std::cout<< prod <<std::endl <<std::endl;
        }   break;
        case 5:
        {   int64_t rem=1;
            for(int u=0;u<i;u++){
                rem%=result[u];
            }
            std::cout<< rem <<std::endl <<std::endl;
        }
        break;
        case 6:
        {   int64_t andv=1;
            for(int u=0;u<i;u++){
                andv&=result[u];
            }
            std::cout<< andv <<std::endl <<std::endl;
        }   break;
    }
    enterContinues();
}

void readFractions(){
    std::cout << std::endl << "--Performing Floating-Point Calculations--" <<std::endl;
    std::cout << "Float_value= max_n / formula" << std::endl;
```

```
formInstr();
int sel=0, i=0;
std::string value;
std::cout << "Select a formula from set(1-10)>>";
getline(std::cin, value);
sel=strtol(value.c_str(), 0, 10);

std::cout << std::endl<< "Enter the number of iterations>>";
std::string vale;
getline(std::cin, vale);
i=strtol(vale.c_str(), 0, 10);
float result[i];

enterStarts();
switch(sel){
    case 1:
        for(int l=0;l<=i;l++) {
            result[l]=1/(45*l);
        }
        break;
    case 2:
        for(int l=0;l<=i;l++) {
            result[l]=1/(20*l+5);
        }
        break;
    case 3:
        for(int l=0;l<=i;l++) {
            result[l]=1/(15*l);
        }

        break;
    case 4:
        for(int l=0;l<=i;l++) {
            result[l]=1/(11*l+1);
        }

        break;
    case 5:
        for(int l=0;l<=i;l++) {
            result[l]=1/(9*l);
```

```
        }

        break;
    case 6:
        for(int l=0;l<=i;l++) {
            result[l]=1/(7*l+3);
        }
        break;
    case 7:
        for(int l=0;l<=i;l++) {
            result[l]=1/(6*l+3);
        }

        break;
    case 8:
        for(int l=0;l<=i;l++) {
            result[l]=1/(5*l+5);
        }

        break;
    case 9:
        for(int l=0;l<=i;l++) {
            result[l]=1/(5*l);
        }
        break;
    case 10:
    { std::cout<<std::endl<<"--Missing Float-Number Calculations--" << std::endl;
      for(int l=0;l<=i;l++) {
          result[l]=1/(1 + pow(l, 2.0) + pow(l, 6.0) + pow(l, 7.0));
      }
    }   break;
    default:
        std::cout << "Selection cannot be found"<< std::endl;
        break;
}
enterStarts();
std::cout <<std::endl<< "    Float Iterations on Formula "<< sel << std::endl;
std::cout << "================================"<< std::endl;
std::cout << "n            float-point result"<< std::endl;
for(int p=0;p<i;p++){
```

```
        std::cout <<""<<p<<"              "<<std::fixed<<std::setprecision(8)<<result[p]<< std::endl;
        if(p%10==0){
            enterContinues();
            if(getGO()>1) break; else continue;
        }
    }
    std::cout << "================================="<< std::endl;


}

void readDraw(){
    std::cout <<std::endl<< "----MEMCELL Draws----" << std::endl;

    std::cout<<"Which integer sequence type? (1)1-9 or (2)1-17 (3)AIS (4)read (5)index>>";
    std::cout<<std::endl;
    std::string sel;
    getline(std::cin, sel);
    int selv=strtol(sel.c_str(), 0, 10);
    switch(selv){
        case 1: {
            std::cout<<std::endl<<"MEMCELL ARRAY:INT";
            for(int j=1;j<=9;j++){
                sel=j< 0? "-":"+";
                std::cout<<std::endl<<sel <<" :: ";
                int k=j;
                while(k>0){
                    std::cout<< "0";
                    k--;
                }

            }std::cout<< std::endl;
        }break;
        case 2: {
            std::cout<<std::endl<<"MEMCELL ARRAY:DIV";
            for(int j=0;j<5;j++){
                sel=divs[j]< 0? "-":"+";
                std::cout<<std::endl<<sel <<" :: ";
                int k=divs[j];
                while(k>0){
```

```
            std::cout<< "0";
            k--;
        }

    }std::cout<< std::endl;
}break;
case 3: {
    std::cout<<"How many iterations in MEM?>>" ;
    std::cout<<std::endl;
    getline(std::cin, sel);
    selv=strtol(sel.c_str(), 0, 10);
    std::cout<<std::endl<<"MEMCELL ARRAY:AIS";
    for(int j=0;j<selv;j++){
        sel=ansi[j]< 0? "-":"+";
        std::cout<<std::endl<<sel <<"::";
        int k=abs(ansi[j]);
        while(k>0){
            std::cout<< "0";
            k--;
        } std::cout<<"#"<< std::endl;
    }std::cout<< std::endl;
}
break;
case 4: {
    std::string data;
    getline(std::cin, data);
    int colpos=data.find("::");
    int haspos=data.find("#");
    int value=haspos-colpos;
    std::string sign=data.substr(0, 1);
    std::string dotcell=data.substr(4, haspos-2);

    int size=dotcell.length();
    std::cout<< "Data cell::"<<data<<" from MEMCELL ---==="<<sign<< size <<std::endl;
}   break;
case 5: {
    std::cout<<"Give index in MEMARRAY?>>" ;
    std::cout<<std::endl;
    getline(std::cin, sel);
    selv=strtol(sel.c_str(), 0, 10);
```

```
            int mem=ansi[selv];
            sel=mem< 0? "-":"+";
            std::cout<<std::endl<<sel <<"::";
            int k=abs(mem);
            while(k>0){
                std::cout<< "0";
                k--;
            } std::cout<<"#"<< std::endl;
            std::cout<< std::endl;

        }break;
    }
}

void setFieldAxiom(){
    std::cout<<"    Setting the Field Axioms" <<std::endl;
    std::cout<<"================================"<<std::endl;

}

void readFieldAxiom(){

}

void *setSqrtValues(void* arg){
    std::cout<<"(RDSC)Setting the Square Root(SQRT) Values" <<std::endl;
    std::cout<<"=========================================="<<std::endl;
    for(int i=0;i<enumSize;i++){
        sqrtVec.push_back(sqrt(ansi[i]));
    }
    std::cout<<"(RDSC) Setting SQRT [Ends]"<<std::endl;
}

void readSqrtValues(){
    std::cout<<"(RDSC)Accessing the Square Root(SQRT) Values" <<std::endl;
    std::cout<<"============================================"<<std::endl;
    std::cout<<"Element         Sqrt"<<std::endl;
    for(int i=0;i<enumSize;i++){
        std::cout<<ansi[i]<<"        "<<sqrtVec[i]<<std::endl;
```

```
        // sqrtVec.pop_back();

        if(i%8==0){
            enterContinues();
            if(getGO()>1) break; else continue;
        }
        //sqrtVec.pop_back();
    }
    std::cout<<"(RDSC) Accessing SQRT [Ends]"<<std::endl;

}

void createSqrtProc(){
    pthread_t* sqrT;
    int reqrt=pthread_create(sqrT, NULL, &setSqrtValues, NULL);
    if(reqrt!=0){
        printf("Error: Setting square root values");
    }
    // pthread_exit(NULL);
}

void *setSqrt1Values(void* arg){
    std::cout<<"(RDSC)Setting the Square Root(SQRT+1) Values" <<std::endl;
    std::cout<<"========================================="<<std::endl;
    for(int i=0;i<(enumSize);i++){
        sqrt1Vec.push_back(sqrt(ansi[i]+1));
    }
    std::cout<<"(RDSC) Setting SQRT+1 [Ends]"<<std::endl;
}

void readSqrt1Values(){
    std::cout<<"(RDSC)Accessing the Square Root(SQRT+1) Values" <<std::endl;
    std::cout<<"==========================================="<<std::endl;
    std::cout<<"Element         Sqrt"<<std::endl;

    for(int i=0;i<enumSize;i++){
        std::cout<<ansi[i]<<"        "<<sqrt1Vec[i]<<std::endl;
        if(i%8==0){
            enterContinues();
            if(getGO()>1) break; else continue;
```

```
        }
        //sqrtVec.pop_back();
    }
    std::cout<<"(RDSC) Accessing SQRT+1 [Ends]"<<std::endl;

}

void createSqrt1Proc(){
    pthread_t* sqrT;
    int reqrt=pthread_create(sqrT, NULL, &setSqrt1Values, NULL);
    if(reqrt!=0){
        printf("Error: Setting square root values 1");
    }
    // pthread_exit(NULL);
}

void *setSqrt6Values(void* arg){
    std::cout<<"(RDSC)Setting the Square Root(SQRT-6) Values" <<std::endl;
    std::cout<<"=========================================="<<std::endl;
    for(int i=0;i<enumSize;i++){
        sqrt6Vec.push_back(sqrt(ansi[i]-6));
    }
    std::cout<<"(RDSC) Setting SQRT-6 [Ends]"<<std::endl;
}

void readSqrt6Values(){
    std::cout<<"(RDSC)Accessing the Square Root(SQRT-6) Values" <<std::endl;
    std::cout<<"=========================================="<<std::endl;
    std::cout<<"Element         Sqrt"<<std::endl;

    for(int i=0;i<enumSize;i++){
        std::cout<<ansi[i]<<"        "<<sqrt6Vec[i]<<std::endl;
        if(i%8==0){
            enterContinues();
            if(getGO()>1) break; else continue;
        }
        //sqrtVec.pop_back();
    }
    std::cout<<"(RDSC) Accessing SQRT-6 [Ends]"<<std::endl;
```

```
}

void createSqrt6Proc(){
    pthread_t* sqrT;
    int reqrt=pthread_create(sqrT, NULL, &setSqrt6Values, NULL);
    if(reqrt!=0){
        printf("Error: Setting square root -6 values");
    }
    // pthread_exit(NULL);
}

void *setSqrt2XnValues(void* arg){
    std::cout<<"(RDSC)Setting the Square Root(SQRT+2Xn) Values" <<std::endl;
    std::cout<<"=========================================="<<std::endl;
    for(int i=0;i<enumSize;i++){
        sqrt2Vec.push_back(sqrt(ansi[i]+2));
    }
    std::cout<<"(RDSC) Setting SQRT+2Xn [Ends]"<<std::endl;
}

void readSqrt2XnValues(){
    std::cout<<"(RDSC)Accessing the Square Root(SQRT+2Xn) Values" <<std::endl;
    std::cout<<"=========================================="<<std::endl;
    std::cout<<"Element         Sqrt"<<std::endl;

    for(int i=0;i<enumSize;i++){
        std::cout<<ansi[i]<<"        "<<sqrtVec[i]<<std::endl;
        if(i%8==0){
            enterContinues();
            if(getGO()>1) break; else continue;
        }
        //sqrtVec.pop_back();
    }
    std::cout<<"(RDSC) Accessing SQRT+2Xn [Ends]"<<std::endl;

}

void createSqrt2XnProc(){
    pthread_t* sqrT;
    int reqrt=pthread_create(sqrT, NULL, &setSqrt2XnValues, NULL);
```

```
    if(reqrt!=0){
        printf("Error: Setting square root -2Xn values");
    }
    // pthread_exit(NULL);
}

void datagridfunc(){
    std::cout<<"   Data Grid"<<std::endl;
    std::cout<<"================="<<std::endl;
    enterStarts();
    int* val=getFunctValues("Enter the number of rows and columns:Format[row,col]>>");
    std::cout<<std::endl;
    for(int j=0;j<val[1];j++){
        for(int i=0;i<val[0];i++){
            if(abs(datagrid[j][i])<10)
                std::cout<<datagrid[j][i]<<"   ";
            else if(abs(datagrid[j][i])>=10 && abs(datagrid[j][i])<=99)
                std::cout<<datagrid[j][i]<<"  ";
            else if(abs(datagrid[j][i])>=100 && abs(datagrid[j][i])<=999)
                std::cout<<datagrid[j][i]<<" ";
            else
                std::cout<<datagrid[j][i];
        }
        std::cout<<std::endl;

    }
    enterContinues();
}

void datacube(){
    std::cout<<"   Data Cube"<<std::endl;
    std::cout<<"================="<<std::endl;
    enterStarts();
    int k=0;
    for(int j=0;j<m;j++){
        for(int i=0;i<n;i++){
            datagrid[j][i]=ansi[k++];
        }
    }
    std::cout<<std::endl;
```

```
    for(int j=0;j<12;j++){
        for(int i=0;i<6;i++){
            if(abs(datagrid[j][i])<10)
                std::cout<<datagrid[j][i]<<"   ";
            else if(abs(datagrid[j][i])>=10 && abs(datagrid[j][i])<=99)
                std::cout<<datagrid[j][i]<<"  ";
            else if(abs(datagrid[j][i])>=100 && abs(datagrid[j][i])<=999)
                std::cout<<datagrid[j][i]<<" ";
            else
                std::cout<<datagrid[j][i];
        }
        std::cout<<std::endl;

    }
    enterContinues();
}

void datahand(){
    std::cout<<"   Data Hand"<<std::endl;
    std::cout<<"================="<<std::endl;
    enterStarts();
    std::cout<<std::endl;
    for(int j=0;j<15;j++){
        for(int i=0;i<5;i++){
            if(abs(datagrid[j][i])<10)
                std::cout<<datagrid[j][i]<<"   ";
            else if(abs(datagrid[j][i])>=10 && abs(datagrid[j][i])<=99)
                std::cout<<datagrid[j][i]<<"  ";
            else if(abs(datagrid[j][i])>=100 && abs(datagrid[j][i])<=999)
                std::cout<<datagrid[j][i]<<" ";
            else
                std::cout<<datagrid[j][i]<<" ";
        }
        std::cout<<std::endl;
    }
    enterContinues();
}

void createImage(){
    std::cout<< "---Data Image"<<std::endl;
```

```
        int* ind=getFunctValues("Enter the indexes of data vector[a,b]>>");
        std::ofstream img;
        img.open("ais.ppm");
        img<<"P3"<<std::endl;
        img<<abs(ansi[ind[0]])<<" "<<abs(ansi[ind[1]]) <<std::endl;
        img<<"255"<<std::endl;
        std::cout<<std::endl<<"Data :"<<abs(ansi[ind[0]])<<":=:"<<abs(ansi[ind[1]])<<std::endl;

        for(int i=0;i<abs(ansi[ind[0]]);i++){
            for(int l=0;l<abs(ansi[ind[1]]);l++){
                int r=l%255;
                int g=i%255;
                int b=(i*l)%255;
                img<<r<<" "<<g<<" "<<b <<std::endl;
            }
        }
        img.close();
        system("open ais.ppm");
    }

}


int main(int argc, char** argv) {

    sleep(1);
    std::string con="-----------------------------------------------";
    std::cout << con;

    // system("clear");
    std::cout <<std::endl<< "--Offline Data Organization and Storage(ODOS)--" << std::endl;
    sleep(1);
    std::cout << "Dataware System: by Dr. F. Appiah " << std::endl;
    std::cout << "Data Operations and Representation Structures" << std::endl;
    std::cout << "       Appiah Integer Sequence" << std::endl;
    sleep(1);
    std::cout << "-----------------------------------------------" << std::endl;
    std::cout << system("date")<< std::endl;
    std::cout << "" << std::endl;
    // alarm(2);
```

```
    sleep(2);
    std::cout << "Running ANS Processor: Reduced Data Set Computer (RDSC)" << std::endl;
    std::cout << "(RDSC) Data is information that can be used in making calculations or decisions or
sequencing events." << std::endl;
    std::cout << "(RDSC) This dataware contains numerical values from ANS program." << std::endl;

    enterStarts();
    std::cout << "    (RDSC) Dataset Menu Order" << std::endl;
    std::cout << "==================================" << std::endl;
    sleep(1);
    std::cout << "(RDSC) Default Array and String Reads" << std::endl;
    std::cout << "(RDSC) 1. ANSVector Read" << std::endl;
    std::cout << "(RDSC) 2. ANSQueue  Read" << std::endl;
    sleep(1);
    std::cout << "(RDSC) 3. ANS-Set   Read" << std::endl;
    std::cout << "(RDSC) 4. ANSList   Read" << std::endl;
    std::cout << "(RDSC) 5. ANS-Formula   Read" << std::endl;
    sleep(1);
    std::cout << "(RDSC) 6. Arithmetics   Read" << std::endl;
    std::cout << "(RDSC) 7. MEMDraw   Read" << std::endl;
    std::cout << "(RDSC) 8. 5-Divisions   Read" << std::endl;
    std::cout << "(RDSC) 9. Magic No. Read" << std::endl;
    sleep(1);
    std::cout << "(RDSC) 10.Floating-point Read" << std::endl;
    std::cout << "(RDSC) 11.Cardinal Numbers Read" << std::endl;
    std::cout << "(RDSC) 12.Examine Data Read" << std::endl;
    std::cout << "(RDSC) 13.Limit Data Read" << std::endl;
    std::cout << "(RDSC) 14.Data Visualization Read" << std::endl;

    std::cout << "(RDSC) Close/Quit   Reading(Ctrl+c)" << std::endl;
    std::cout << "============================" << std::endl;

    std::cout << "Do enter number no(2) after ANS-Formula" << std::endl << "or else have wait on the next loop
to quit." << std::endl << std::endl;
    sleep(1);
    setIntANS();
    std::cout << "Setting Appiah Integer Sequence....." << std::endl;
    setANSMap();
    std::cout << "Setting Appiah Long Sequence....." << std::endl;
    sleep(1);
```

```
    setANSQueue();
    std::cout << "Setting Appiah Double Sequence....." << std::endl << std::endl;
    setANSList();
    setANSet();
    createSqrtProc();
    createSqrt1Proc();
    createSqrt6Proc();
    createSqrt2XnProc();
    // setANString();
    setANSVector();
    setMagicNumber();
    std::cout << "Accessing data structures in store[MEM]....." << std::endl << std::endl;
    enterStarts();
    std::cout << "Reading datarray [MEMARRY]:FIFO....." << std::endl << std::endl;

    for(int k=0;k<getEnumSize();k++){
        std::cout << "Accessing [index "<< k <<"]....." << getIntANS()[k] << std::endl;
        usleep(995);
    }
//    for(int k=0;k<getEnumSize();k++){
//        std::cout << getANString()[k] << std::endl << std::endl;
//    }
//reading from the store structures
    bool stop= false;
    while(!stop) {
        if(getGO()<2) {
            readANSVector();
            std::cout << system("time")<< std::endl;
            readANSQueue();
            std::cout << system("time")<< std::endl;
            readANSet();
            std::cout << system("time")<< std::endl;
            readANSList();
            std::cout << system("time")<< std::endl;
            readFormula();
            std::cout << system("time")<< std::endl;
            readArith();
            std::cout << system("time")<< std::endl;
            readDraw();
            std::cout << system("time")<< std::endl;
```

```
            readMN45();
            std::cout << system("time")<< std::endl;
            readFractions();
            std::cout << system("time")<< std::endl;
            read5Divs();
            std::cout << system("time")<< std::endl;
            getCardinalNumbers();
            std::cout << system("time")<< std::endl;
            examineData();
            std::cout << system("time")<< std::endl;
            readSqrtValues();
            std::cout << system("time")<< std::endl;
            readSqrt1Values();
            std::cout << system("time")<< std::endl;
            readSqrt6Values();
            std::cout << system("time")<< std::endl;
            readSqrt2XnValues();
            std::cout << system("time")<< std::endl;
            datacube();
            datagridfunc();
            datahand();
            std::cout << system("time")<< std::endl;
            createImage();
            std::cout << system("time")<< std::endl;

        }
        else stop=true;
    }
    uninitialise();
    pthread_exit(NULL);
    std::cout << "(RDSC)::Running was successful....." << std::endl;

    return (EXIT_SUCCESS);
}
```

BUDGET

Project budget estimated at cost of GBP 1M.

Description	Quantity	Unit Price	Cost
Project Cost	95	GH₵10,000	GH₵950,000
Research Fee	40	GH₵90,000	GH₵3,600,000
Computer Equipment	20	GH₵500,000	GH₵10,000,000
Total			**GH₵14,550,000**